Subject: Computer Organization and Architecture

Password validity checking system

Class: SY-ETD
Batch: B1
Group No: 02

Roll number	Name	PRN
4	Sahil Parekh	12010105
6	Sakshi Kulkarni	12010070
10	Sarthak Bhake	12010005
13	Sejal Sayam	12010928
14	Shajjad Shaikh	12010659

Department Of Electronics & Telecommunication

Vishwakarma Institute of Technology, Pune

A.Y. 2021-22 Sem 2 SY

Introduction and Problem Statement:

This is a "**Password Validity checking system**" project which is built on the Intel 8086 microcontroller. This project is executed using TASM and DosBox. With the help of this project we were able to brush up our basics of assembly language programming. Various different parameters like registers , segments , pointers which we learned in our theory were seen in action while we did this project. The code is very trivial to understand as it is modeled in a very logical flow which is understandable to even a layman excluding the syntaxes. The problem statement of this project is to write a program in 8086 Assembly Language that checks an input string against a password string stored in memory and outputs a message based on whether the password is matched or not.

Tools Used:

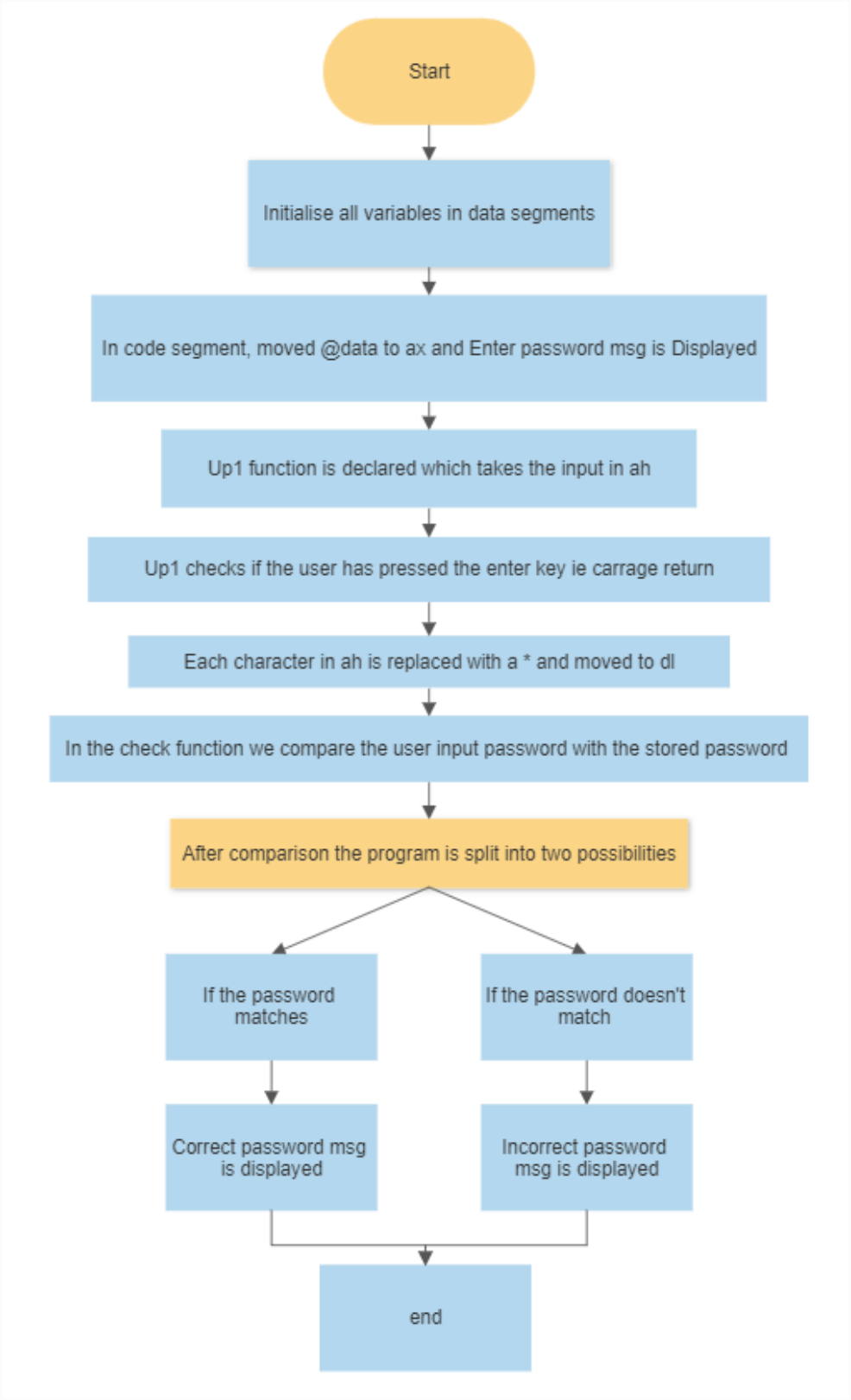
1. DOSBox

DOSBox is a free and open-source emulator which runs software for MS-DOS compatible disk operating systems—primarily video games. It was first released in 2002. DOSBox is a command-line program, configured either by a set of command-line arguments or by editing a plain text configuration file. For ease of use, several graphical front ends have been developed by the user community.

2. TASM 1.4

Tasm, or Turbo Assembler/Turbo Debugger By Borland, is one of the most popular assemblers used in the world today. Tasm Windows 7 - Windows 8 64-bit By Techapple.net is developed by Techapple.Net and is used by 2 users of Software Informer. The name of the program executable file is DOSBox.exe. This particular product is not fit to be reviewed by our informers. Tasm Windows 7 - Windows 8 64 bit By Techapple.net is developed by Techapple.Net and is used by 2 users of Software Informer. The name of the program executable file is DOSBox.exe. This particular product is not fit to be reviewed by our informers.

Flowchart of the Project:



Instructions used in the program:

.model small - The memory model directive specifies the size of memory which the program needs. (small- it can't be more than 64K) in our case the program is under 64k

.stack 100h - Segment directive which defines 100h words as program STACK.

.**data** - These instructions are used to transfer the data from source operand to destination operand.

password db 'coaa' - In assembly language, we use "db" (define byte) to allocate some space, and fill it with a string.

len equ ($-password) - length of string. $ It states the end of a Statement.

10 dup(0) - dup tells assembler to duplicate an expression 0 number of times here

.code - It is where the executable program is stored.

mov ax,@data - loading starting address of data segment in ax

lea dx,msg1 - Prints the message “Enter Your Password:”

mov ah,09h - output message

int 21h - means call the interrupt handler

mov si, 00 - moving source index to 00

mov ah,08h - it reads the 8 bit char from the input device without echoing it to output device

cmp al,0dh - compares with ASCII 13 which is carriage return

je down - jump if equal

mov [inst+si],al - the al register points to si^{th} element of the inst array

mov dl,'*' - moves the symbol asterisk (*) to register dl

mov ah,02h - request display character

inc si - incrementing the value of source index register by 1

jmp up1 - unconditional jump. Such an instruction transfers the flow of execution by changing the program counter.

mov bx,00 - Moves 0 to the bx register

mov cx,len - Moves the length of the password (i.e 4) to the cx register

mov al,[inst+bx] - The bx^{th} element of the inst array is moved to the al register

mov dl,[password+bx] - The bx^{th} element of the password array is moved to the dl register

cmp al,dl - This compares the user input password and the password already stored in the memory

jne fail - If the comparison is not successful the program flow jumps to the fail function where the error message is displayed.

inc bx - incrementing bx to point to the next element

loop check - looping the check function

lea dx,msg2 - Prints the message "Correct Password,Welcome!"

jmp finish - an unconditional jump from current location to new location, finish in this case

lea dx,msg3- Prints the message **"Incorrect Password, Please try again"**

mov ah,4ch - store or move the hexadecimal value 4C into register ah in order to exit the program

end - end of the program

Screenshot of the Program:

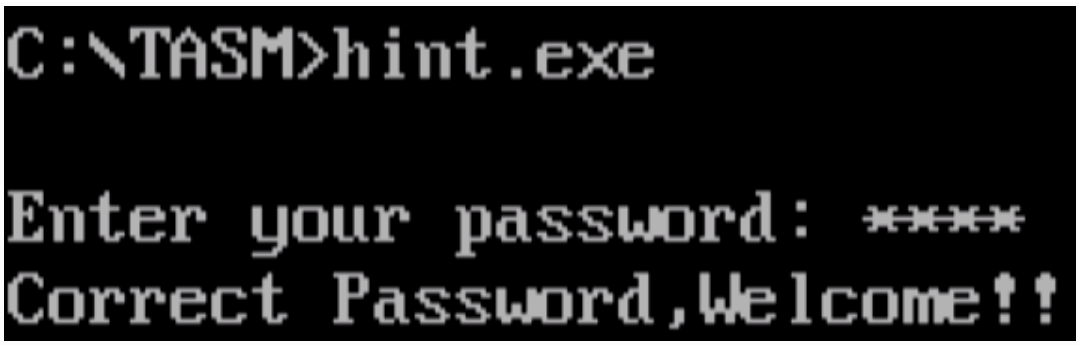
```
S.model small
.stack 100h

.data
password db 'coaa'
len equ ($-password)
msg1 db 10,13,'Enter your password: $'
msg2 db 10,13,'Correct Password,Welcome!!$'
msg3 db 10,13,'Incorrect Password, Please try again!$'
new db 10,13,'$'
inst db 10 dup(0)

.code

start:
mov ax,@data
mov ds,ax
lea dx,msg1
mov ah,09h
int 21h
mov si,00
```

```
up1:
mov ah,08h
int 21h
cmp al,0dh
je down
mov [inst+si],al
mov dl,'*'
mov ah,02h
int 21h
inc si
jmp up1


down:
mov bx,00
mov cx,len

check:
mov al,[inst+bx]
mov dl,[password+bx]
cmp al,dl
```

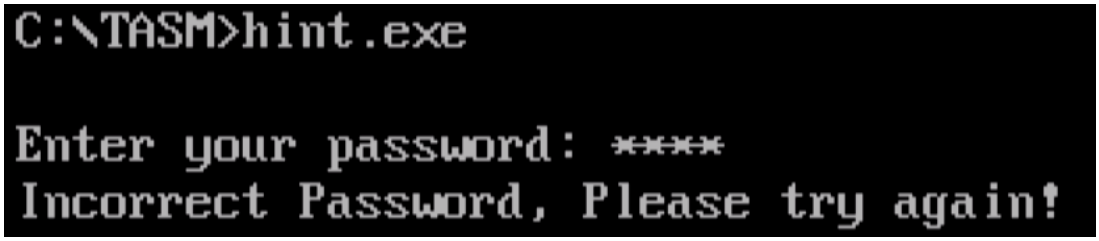
```
jne fail
inc bx
loop check
lea dx,msg2
mov ah,09h
int 21h
jmp finish

fail:
lea dx,msg3
mov ah,09h
int 21h

finish:
mov ah,4ch
int 21h


end start
end
```

Screenshot of the Output:

Case 1: Correct Password



Case 2: Incorrect Password



Conclusion:

- The given password in the data segment is successfully checked and compared with the input password entered by the user.
- The result of the above operation is displayed using appropriate messages after giving input by user.
- Each time program is terminated with exit code 0 properly without any error.